What is claimed is:

1. A method for holding up R-unit operands for a minimum number of cycles until all prior updates have completed by comparing addresses in at least one queue and interlocking valid R-unit register address matches, the method comprising:

receiving a plurality of R-unit register addresses;

storing said R-unit register addresses in a plurality of queues;

accessing said queues;

comparing said R-unit register addresses;

determining matches between R-unit register addresses; and

implementing one or more interlocks after said determining a valid match.

2. The method of claim 1 wherein said interlock causes a read instruction not to execute.

3. The method of claim 1 wherein said plurality of queues includes a write queue, pre-write queue and a read queue.

4. The method of claim 3 wherein a bypass sends an R-unit register addresses when said read queue is empty.

5. The method of claim 3 wherein said comparing includes comparing said R-unit register addresses sent to said read queue against said R-unit register addresses sent to said write queue.

6. The method in claim 3 wherein said determining includes matching a valid R-unit register addresses of said write queue and said read queue.

7. The method in claim 3 wherein said determining includes matching said valid R-unit register addresses of said pre-write queue and said read queue.

8. The method in claim 1 wherein said interlocks are implemented after said valid R-unit register address match is determined.

9. The method in claim 1 wherein said interlocks prevent read instructions from being processed.

10. The method in claim 1 wherein said write queue accumulates R-unit register addresses.

11. The method in claim 1 wherein said updating occurs when an SRAM receives the accumulated results from said write queue.

12. A system for holding up R-unit operands for a minimum number of cycles until all prior updates have completed by comparing R-unit register addresses in at least one queue and interlocking valid R-unit register address matches, the system comprising:

a plurality of queues for storing R-unit register addresses;

a comparator for comparing said R-unit register addresses in said plurality of queues and determining matches between R-unit register addresses; and

a plurality of interlocks that are implemented after determining valid matches of said R-unit register addresses.

13. The system of claim 12 wherein one of said interlocks causes a read instruction not to execute.

14. The system of claim 12 wherein said plurality of queues includes a write queue, a pre-write queue, and a read queue.

15. The system of claim 14 further comprising a bypass that sends an R-unit register addresses when said read queue is empty.

16. The system of claim 14 wherein said comparator compares said R-unit register addresses sent to said read queue against said R-unit register addresses sent to said write queue.

17. The system in claim 15 wherein said comparator determines said valid R-unit register address matches between said write queue and said read queue.

18. The system in claim 15 wherein said comparator determines said valid R-unit register address matches between said pre-write queue and said read queue.

19. The system in claim 13 wherein said interlocks are implemented after said valid R-unit register address match is determined.

20. The system in claim 13 wherein said interlocks prevent read instructions from being processed.

21. The system in claim 14 wherein said updating allows R-unit register addresses to accumulate in said write queue.

22. The system in claim 14 wherein said R-unit is updated when an SRAM receives the accumulated results from said write queue.